LETS GROW MORE

NAME - VIVEK SHARMA

DATA SCIENCE INTERN

TASK-1: Iris Flower Classification ML Project

Description of Task:

This particular ML Project is usually referred to as the "Hello World" of machine learning. The Iris flowers dataset contains numeric attributes , and it is perfect for beginners to learn about supervised ML atlgorithms and to empower themself in the field of data science, learn how to load and handle data. Also, since this is a small dataset,it can easily fit in memory without requiring special transformation or scaling capabilies.

Datasets: http://archive.ics.uci.edu/ml/datasets/iris

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as mat
import seaborn as ss
import scikitplot as skplt
from sklearn.datasets import load_iris
```

In [2]:
```
df = pd.read_csv('D:\da\Iris.csv')#names=["sepalLength","sepalWidth","petalLength","petalWidth","class"])
df.head()
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
df.tail()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [4]:
```
df.info() #getting information about data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [5]:
```
df.shape
```

Out[5]: (150, 6)

In [6]:
```
df.Species.nunique()
```

Out[6]: 3

In [7]:
```
df.isnull() #checking if any value is null
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

In [8]:
```
df.describe() #display stats about the iris data
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [9]:
```
df.max()
```

Out[9]:
```
Id                          150
SepalLengthCm               7.9
SepalWidthCm                4.4
PetalLengthCm               6.9
PetalWidthCm                2.5
Species          Iris-virginica
dtype: object
```

In [10]:
```
df.min()
```

Out[10]:
```
Id                         1
SepalLengthCm            4.3
SepalWidthCm             2.0
PetalLengthCm            1.0
PetalWidthCm             0.1
Species          Iris-setosa
dtype: object
```

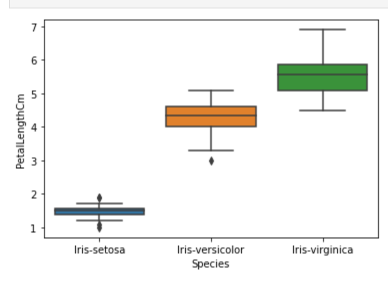
In [11]:
```
df.corr()
```

Out[11]:

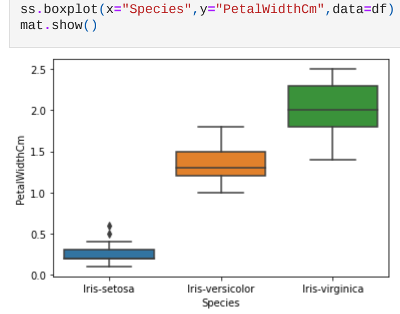
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

DATA VISUALIZATION

In [12]:
```
#the boxplot is rated with boxplot() method. the example below loads the flower data set
#the representation below shows the minimum, maximum, median, 1st quartile and 3rd quartile
ss.boxplot(x="Species",y="PetalLengthCm",data=df)
mat.show()
```

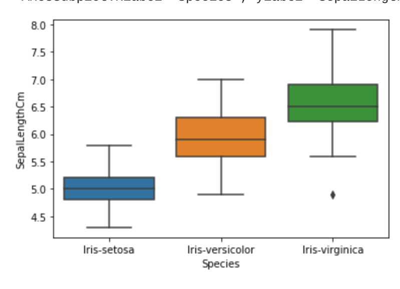


In [13]:
```
ss.boxplot(x="Species",y="PetalWidthCm",data=df)
mat.show()
```



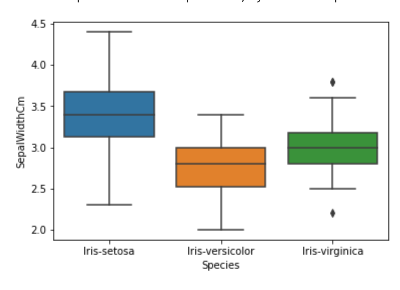
In [14]:
```
ss.boxplot(x="Species",y="SepalLengthCm",data=df)
```

Out[14]: <AxesSubplot:xlabel='Species', ylabel='SepalLengthCm'>



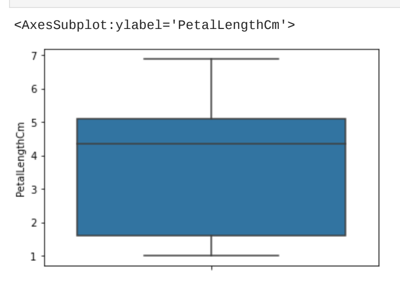
In [15]:
```
ss.boxplot(x="Species",y="SepalWidthCm",data=df)
```

Out[15]: <AxesSubplot:xlabel='Species', ylabel='SepalWidthCm'>



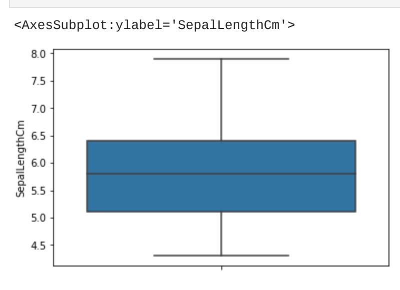
In [16]:
```
ss.boxplot(y="PetalLengthCm",data=df)
```

Out[16]: <AxesSubplot:ylabel='PetalLengthCm'>



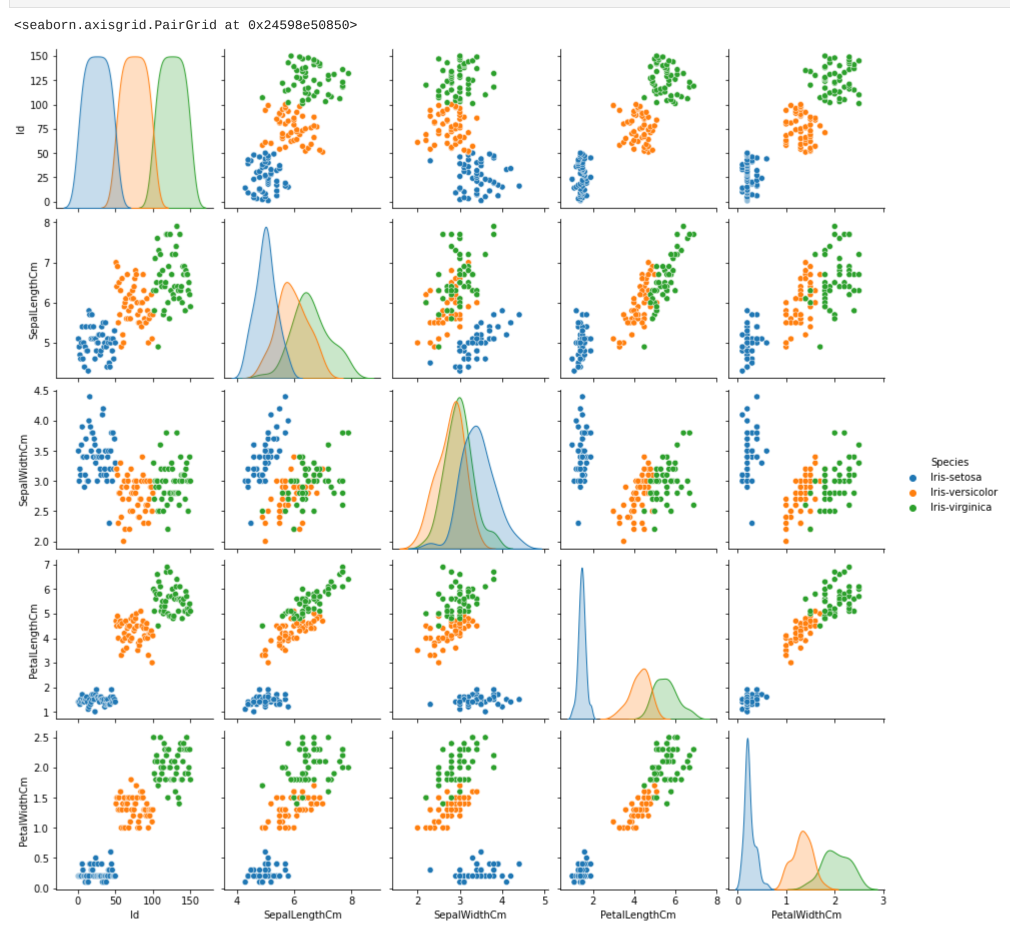
In [17]:
```
ss.boxplot(y="SepalLengthCm",data=df)
```

Out[17]: <AxesSubplot:ylabel='SepalLengthCm'>



In [18]:
```
ss.pairplot(df,hue="Species")# pairwise relationship in the dataset between different attributes
```

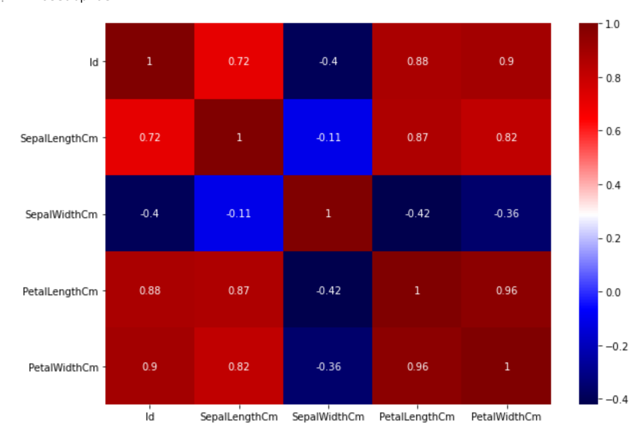
Out[18]: <seaborn.axisgrid.PairGrid at 0x24598e50850>



DATA PREPROCESSING AND CORRELATION MATRIX

In [19]:
```
#heatmap uses to show 2D data in graphical format
#Each data value represents in a matrix and it has a special color
#'true' value to annot then the value will show on each cell of the heatmap
import matplotlib.pyplot as mat
import seaborn as ss
mat.figure(figsize=(10,7))
ss.heatmap(pd.read_csv('D:\da\Iris.csv').corr(),annot=True,cmap="seismic")
```

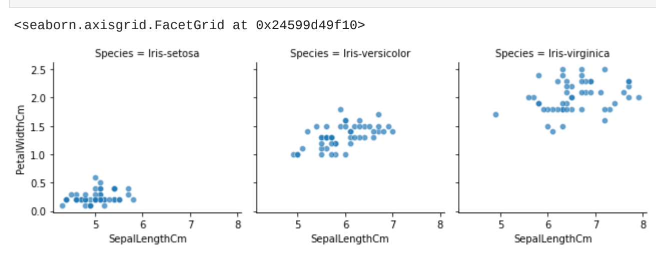
Out[19]: <AxesSubplot:>



SCATTER PLOT

In [20]:
```
grid = ss.FacetGrid(df, col="Species")
grid.map(ss.scatterplot, "SepalLengthCm","PetalWidthCm", alpha=.7);
grid.add_legend()
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x24599d49f10>



LABEL ENCODER

In [21]:
```
from sklearn.preprocessing import LabelEncoder
ll = LabelEncoder() #label encoder can be used to formalise labels
```

In [22]:
```
df["Species"] = ll.fit_transform(df["Species"])
df.head()
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

In [23]:
```
x = df.drop(columns=["Species"]) #drop the column
y = df["Species"]
```

In [24]:
```
x[:5] #return list from the beginning unto index 5
```

Out[24]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2

In [25]:
```
y[:5]
```

Out[25]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
```

Splitting the dataset into Training data and Test data

In [26]:
```
import pandas as pd
array = pd.read_csv('D:\da\Iris.csv').values
from sklearn.model_selection import train_test_split
x = array[:,0:4]
y = array[:,4]
y = y.astype('int')
x_train, x_test , y_train , y_test = train_test_split(x,y,random_state=1,test_size=0.3)
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

In [28]:
```
lr = LogisticRegression()
svc =SVC()
knc = KNeighborsClassifier()
gu = GaussianNB()
rfc = RandomForestClassifier()
dtc = DecisionTreeClassifier()
```

Training and Evaluating the models

In [29]:
```
models = [lr,svc,knc,gu,rfc,dtc]
scores =[]

for model in models:
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    scores.append(accuracy_score(y_test, y_pred))
    print("Accuracy of "+type(model).__name__+" is ",accuracy_score(y_test, y_pred))
```

```
C:\Users\Vivek Sharma\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy of LogisticRegression is  0.8888888888888888
Accuracy of SVC is  0.8666666666666667
Accuracy of KNeighborsClassifier is  0.8666666666666667
Accuracy of GaussianNB is  0.9333333333333333
Accuracy of RandomForestClassifier is  0.8666666666666667
Accuracy of DecisionTreeClassifier is  0.8444444444444444
```

SPLITTING THE DATASETS

In [30]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [31]:
```
df = pd.read_csv('D:\da\Iris.csv')
train, test =train_test_split(df, test_size=0.2, random_state=0)
print("print shape of training data : ",train.shape)
print("print shape of testing data : ",test.shape)
```

```
print shape of training data :  (120, 6)
print shape of testing data :  (30, 6)
```

In [32]:
```
train_x = train.drop(columns=['Species'],axis=1)
train_y = train['Species']

test_x = test.drop(columns=['Species'],axis=1)
test_y = test['Species']
```

In [33]:
```
results = pd.DataFrame({"Models":["Logistis Regression", "K-Nearest Neighbors", "Support Vector Machine", "Naive Bayes",
                        "Decision Tree", " Random Forest"], "Accuracy":scores})
results = results.sort_values(by="Accuracy" , ascending=False)
print(results)
```

```
                  Models  Accuracy
3            Naive Bayes  0.933333
0    Logistis Regression  0.888889
1    K-Nearest Neighbors  0.866667
2  Support Vector Machine  0.866667
4          Decision Tree  0.866667
5          Random Forest  0.844444
```